Thermal-Aware Frequency Scaling for Adaptive Workloads on Heterogeneous MPSoCs

Heng Yu, Rizwan Syed, Yajun Ha
National University of Singapore
Email: {eleyuhen, syed.rizwan, elehy}@nus.edu.sg

***Abstract*—For applications featuring adaptive workloads, the quality of their task execution can be dynamically adjusted given the runtime constraints. When mapping them to heterogeneous MPSoCs, it is expected not only to achieve the highest possible execution quality, but also meet the critical thermal challenges from the continuously increasing chip density. Prior thermal management techniques, such as Dynamic Voltage/Frequency Scaling (DVFS) and thread migration, do not take into account the trade-off possibility between execution quality and temperature control. In this paper, we explore the capability of adaptive workloads for effective temperature control, while maximally ensuring the execution Quality-of-Service (QoS). We present a thermal-aware dynamic frequency scaling (DFS) algorithm on heterogeneous MPSoCs, where judicious frequency selection achieves QoS maximization under the temperature threshold, which is converted to the thermal-timing deadline as an additional execution constraint. Results show that our frequency scaling algorithm achieves as large as 31.5% execution cycle/QoS improvement under thermal constraints.**

I. Introduction

Adaptive applications are receiving growing attentions owing to their capability to provide scalable execution quality in reaction to the execution environment. The more execution cycles assigned to an adaptive application, the higher quality it can achieve. One example is the Scalable Video Coding (SVC) scheme in H.264/MPEG-4 standard, which provides customized service quality to accommodate various network and device conditions [1]. The other example is JPEG2000 codec supporting multiple playback resolutions [2]. Rather than simply completing or failing the execution, adaptive applications usually define multiple execution granularity such that a finer-grained version results in better output quality, at the price of increased program cycles, energy, and thermal impact.

Meanwhile, the temperature of the integrated circuits continues as a major concern. Given the shrinking device features and increased amount of workloads running concurrently, thermal impacts to system reliability and performance are significantly augmented in (deep) sub-micron level. As has been reported, 50% of device lifetime can be reduced by increasing the temperature by 10-15°C [3]. Clock skew could also be prominent due to variations on temperature-aware Elmore delay of interconnects [4]. More severely, leakage current goes up dramatically with temperature increases. Even for 65nm technology, heating from 60°C to 80°C leads to 21% increase in leakage current [5]. The leakage-thermal mutual induction phenomenon, a.k.a. thermal-runaway, are more prone to the increasing gate density that facilitates the lateral heat flow between adjacent cores or functional blocks.

Dynamic Thermal Management (DTM) techniques are proposed as a class of microarchitectural and/or OS-level strategies for runtime temperature control, where adjustment mechanisms including runtime thread migration [6]–[8] and DVFS [9]–[13] are applied. Whereas the abovementioned techniques focus on workload manipulation by deciding which speed level or mapped processor the task[1] runs at, the adaptive applications provide an orthogonal management perspective, where the flexibility in execution is able to tradeoff the heat generation. Namely, in case of thermal violation, execution can be stopped sooner than scheduled while the execution quality can still meet a low-level requirement. As the first of its kind, this work combines the available system-level DTM techniques, specifically dynamic frequency scaling, and explores the thermal management capability of the adaptive application model.

Several recent thermal-aware DVFS techniques are found effective in single-processor scenarios, where the "cool" tasks interleave the "hot"[2] tasks to keep the temperature fluctuating below its threshold [11], [12]. However, in the multiprocessor scenarios, the notion of "cool" task calls for redefinition in the presence of shortened thermal conducting paths under increasing chip density, as well as the parasitic thermal-runaway effect. More specifically, whether the task is cool or hot should depend, not only spatially but also temporally, on the thermal activities of adjacent cores. In turn, state-of-the-art neighbor-aware techniques, such as static ILP-based optimization [14] and quasiconvex programming for thermal-aware speed selection [13], are less efficient when applied to the tasks featuring self-adaptability or large variations in execution volume. Even the proactive temperature management methodologies based on offline trained parameters [6], [8] or historical measurements [7] suffer from accuracy for such applications, given their inadequate awareness of adjacent cores.

In this paper, we propose a frequency scaling-based dynamic

[1]In this paper, we use workload and task interchangeably to represent the executables on the processor.

[2]The "hot" or "cool" tasks are defined such that the steady-state temperatures of the tasks are above or below the threshold, respectively, according to static profiling.

978-3-9815370-2-4/DATE14/©2014 EDAA

scheduling algorithm on heterogeneous MPSoC platforms, leveraging the task-level adaptability for thermal violation avoidance. An analytical, lightweight, and accurate temperature prediction method is presented that takes neighbor cores' dynamic thermal activities into account. The adaptive task is then able to terminate flexibly before thermal constraint violation. The timing flexibility of adaptive applications naturally leads to our optimization goal of maximizing the application execution cycles, which implies the maximization of the execution Quality-of-Service (QoS). Compared to state-of-the-art DTM techniques balancing between throughput/deadline and thermal requirements, considering adaptive applications adds more complexity beyond baseline throughput guarantees, but will benefit from additional thermal management freedom. We propose a guided-search based frequency scaling algorithm taking into account the temperature threshold to globally optimize the total workload cycle/QoS over all processors, where temperature threshold is conveniently converted into timing constraints. Results show that, our frequency scaling algorithm achieves as large as 31.5% execution cycle/QoS improvement under thermal constraints.

In the remaining of the paper, we give a motivational example showing how frequency scaling and adaptive workload can help avoid thermal violations while maximizing performance, in Section II. System modeling and problem definition are presented in Section III. Section IV presents the lightweight thermal timing prediction method. Section V describes the frequency scaling algorithm, and Section VI gives the experimental results. The paper is concluded in Section VII.

II. Motivational Example

The focus of this work is to explore how adaptive applications can be used to tolerate the thermal constraints, in addition to other requirements in the real-time embedded scenarios, while still maximizing the execution quality. In this section, we demonstrate the factors that interact with each other to tradeoff application quality optimization, namely the thermal threshold, neighbor thermal interaction, application execution time, system energy, as well as the scheduling overhead. We consider a simple 2×2 identical tiled cores, as shown in Fig. 1, which are able to run at 100MHz and 300MHz. Assuming tiles T_1 and T_3 are already running at 300MHz, now two identical parallel adaptive tasks, τ, are to be dispatched on T_2 and T_4. The scheduling process is invoked to decide the processor frequency and the execution cycles[3] of τ, under the deadline and thermal constraints. For illustration purpose, the system energy constraint is not considered here for the time being. To derive the temperature profile, we use the temperature simulator tool, HotSpot [15], and feed the parameters of the tiled cores as shown in Table I in Section VI.

Fig. 2 shows the temperature traces of T_2 and/or T_4 that run at 300MHz (red filled box with solid line) and 100MHz (blue empty box with solid line), respectively, considering thermal interference from T_1 and T_3. The green dashed line indicates a thermal threshold of 57°C, which is an assumption for illustration purpose. The other two dashed lines indicate the time-cycle monotonic relationship for a fixed frequency. According to the thermal traces, if the tiles run at 300MHz, execution stops at 13.33ms due to thermal threshold violation. The cycle executed, as indicated at point A(13.33, 4), reaches 4E+9 cycles. On the other hand, if frequency scaling is conducted such that tiles run at 100MHz, no thermal violation happens, and the tiles can run till the deadline at 100s reaching 10E+8 cycles, which is shown at point B(100, 10). Assuming that the task τ has adaptive feature, such that more execution cycles can render higher execution quality, frequency scaling at 100MHz is preferred under the thermal and timing constraints. Another interesting observation from Fig. 2 is that the thermal interference from the neighbors can be significant. The filled box with dashed line indicates the temperature trace of T_2 and T_4, by shutting off T_1 and T_3 when running τ. In this way the thermal interference from T_1 and T_3 is experimentally minimized. A difference of 2.7°C is observed, which can significantly impact the frequency results in Fig. 2.

In this work, we aim to efficiently conducting dynamic frequency scaling for adaptive applications in order to optimize the execution cycles under the thermal, timing,

[3]We use execution cycle to represent the application quality. Their monotonically increasing relationship is presented in Section III.

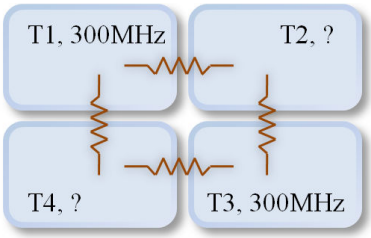

Fig. 1: A four-tiled platform with thermal interaction represented as thermal resistance.

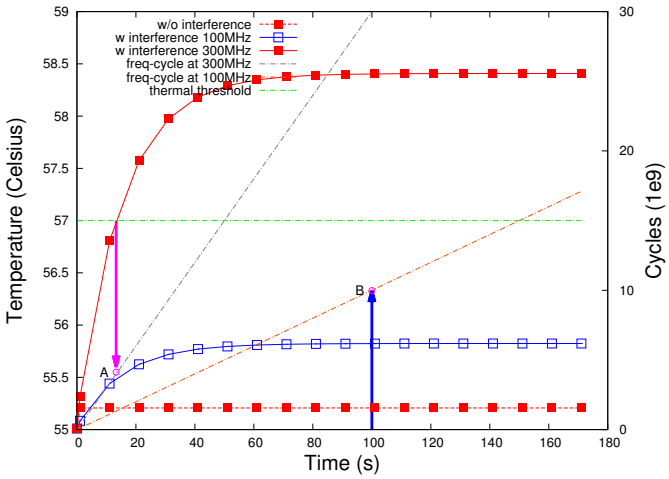

Fig. 2: The illustrative example: frequency scaling leads to cycle increase under the thermal constraint.

and energy constraints. By modeling the neighbor thermal interference effects, we propose an efficient runtime frequency scaling method. As a key step for the optimization, we also propose a fast method to identify the time before thermal violation (as shown in Fig. 2, time to reach point A).

III. System Modeling and Problem Definitions

A. System modeling

1) Adaptive task model: We assume the platform contains a set of N_p heterogeneous processors, including synthesized functional blocks, DSPs, or CPUs that are capable of workload processing. The workload, namely a task t_i, is featured by: (1) average processor switching capacitance $\mathcal{C}_i$ reflecting the processor logic utilization by the task; (2) the execution timing deadline d_i since its invocation; (3) the minimum execution cycles c_i^{min} as the minimum execution requirement to ensure baseline quality requirement q_i^{min}; (4) the ceiling execution cycles c_i^{max} with associated quality q_i^{max}. An important assumption is that the intermediate quality levels in $[q_i^{min}, q_i^{max}]$ increases *monotonically* with execution cycle increase in $[c_i^{min}, c_i^{max}]$, such that the more cycles executed for the workload, the more QoS can be generated by execution. The quality-cycle relationship can be validated by real-life applications, where an example would be the inverse DWT whose data is illustrated in Fig. 3. Please refer to [16] for a list of applications with adaptive features.

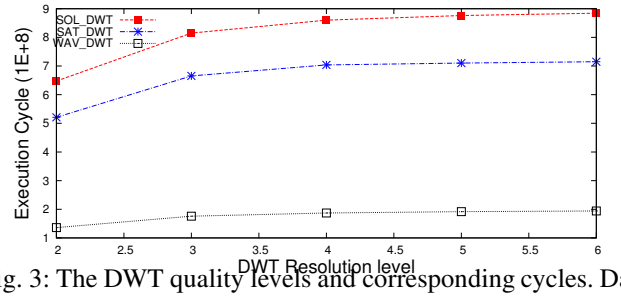

Fig. 3: The DWT quality levels and corresponding cycles. Data obtained by profiling three HD images from NASA.

2) Power model: To realize DVFS, we assume that the processor operates under a finite set of voltage-frequency pairs (v, f_v), where f_v depends on voltage v and is usually set as the highest frequency allowable by v. The total power consumption, P_i, of a task t_i under voltage v consists of both dynamic and leakage components, p_i^{dyn} and p_i^{lkg}. As with [11], [14], we assume the dynamic power does not vary significantly with temperature change, and model it conventionally as $p_i^{dyn} = \mathcal{C}_i v_i^2 f_i$, where we simplify f_{v_i} as f_i. The leakage power changes superlinearly under v_i. According to [17], we adopt the linear model with sufficient accuracy, $p_i^{lkg} = K_1(v_i)T + K_2(v_i)$, where K_1 and K_2 are constants under v_i depending on manufacturing technology, core specification, design style, etc. Then, the total power consumption is

$$P_i = \mathcal{C}_i v_i^2 f_i + K_1(v_i)T + K_2(v_i). \tag{1}$$

Since we focus on the execution cycles of t_i, we define the energy consumption *per cycle*,

$$E_i^{cyc} = \mathcal{C}_i v_i^2 + (K_1(v_i)T + K_2(v_i)) f_i^{-1}. \tag{2}$$

3) Thermal model: We adapt the thermal model from HotSpot, which employs the thermal-electrical duality and computes the processors' temperature with an equivalent lumped thermal RC network [15]. As in [14], we consider (1) the lateral heat conduction, denoted as $G_{m,n}^l$, between adjacent processors m and n; (2) cross-layer heat conduction, G_m^h, between m and the covered heat spreader, h; and (3) thermal capacitance, C_m, for m's transient timing behavior of heat transfer. The temperature of m can then be calculated using the differential equation:

$$\begin{aligned} C_m \frac{dT_m(t)}{dt} = &\sum_{n \in \mathcal{M}} (T_n(t) - T_m(t)) G_{m,n}^l \\ &+ (T_h(t) - T_m(t)) G_m^h + P_{i,m}(T_m(t)), \end{aligned} \tag{3}$$

where $\mathcal{M}$ is the set of neighbor processors of m, and $P_{i,m}(T_m(t))$ is the power of running t_i as a function of $T_m(t)$. Due to higher thermal conductivity of heat spreader (for instance 400W/mK for copper), than 148W/mK for the silicon chip, we treat the heat spreader as one whole unit rather than splitting it into several interactive blocks. Work in [14] specifies the modeling of finer grained heat spreader element interactions, and our model can be readily extended to cover it. The same extension applies to external devices such as heat sinks or fans, which are not included in this work for simplicity of elaboration. We lump the thermal interface material (TIM) together with the heat spreader given its negligible thickness.

4) Problem statement: This work focuses on the dynamic scheduling algorithm design that aims at maximizing the total execution cycles, $\sum_{t_i \in \mathcal{N}} c_i$, of all adaptive tasks t_i in a task set $\mathcal{N}$ which are in the scheduler queue and ready to be dispatched and run in parallel. The optimization is subject to constraints of individual task deadlines d_i, energy budget $E_{\mathcal{N}}$, quality requirements represented by cycles $[c_i^{min}, c_i^{max}]$, and temperature threshold $\mathcal{T}$.

In our approach to deal with the temperature constraint, we convert it into the timing constraint d_i^{thr}, which stands for the time constraint of t_i before it heats above $\mathcal{T}$. Then we present an efficient guided-search based f_i selection heuristic for the $\sum_{t_i \in \mathcal{N}} c_i$ maximization under timing $(min(d_i, d_i^{thr}))$, cycle, and energy constraints.

IV. Deciding Thermal Timing Constraint (d_i^{thr})

The thermal profile of a core m depends on the heating/cooling effects due to self-execution and neighbor conduction. So obtaining $T_m(t)$ requires the temperature profiles of all processors and the heat spreader. In the context of m, we rewrite (3) as

$$\begin{aligned} C \frac{dT(t)}{dt} = &\sum_{n \in neighbor} (g_n(t) - T(t)) \cdot G_n^l \\ &+ (g_n^{sp}(t) - T(t)) \cdot G^h + P_i(T(t)), \end{aligned} \tag{4}$$

where $g_n(t)$ and $g_n^{sp}(t)$ are the temperature of neighbor n and heat spreader, respectively. According to (1), $P_i(T(t)) = \mathcal{C}_i v_i^2 f_i + K_1(v_i)T(t) + K_2(v_i) = K_1'T(t) + K_2'(f_i)$, where under fixed v_i, $K_1' = K_1(v_i)$ is constant, and $K_2' = K_2(v_i) + \mathcal{C}_i v_i^2 f_i$ is linear to f_i. Then, in the standard linear form, (4) becomes

$$\frac{dT(t)}{dt} + \alpha T(t) = \frac{1}{C}(K_2' + \sum_{x\in\{n|sp\}} g_x(t)\cdot G_x), \quad (5)$$

where $G_{x\in\{n|sp\}}$ is the respective conductance from the processor or heat spreader to m, and $\alpha = \frac{\sum_{x\in\{n|sp\}} G_x - K_1'}{C}$. With integrating factor $u(t) = e^{\alpha t}$, we solve (5) as

$$\begin{aligned} u(t)T(t) &= \frac{K_2'}{C}\int u(t)dt + \frac{1}{C}\sum_x (G_x \int u(t)g_x(t)dt) + C_0 \\ &= \frac{K_2'}{\alpha C}u(t) + \frac{1}{C}\sum_x (G_x \int u(t)g_x(t)dt) + C_0', \end{aligned} \quad (6)$$

where $C_0' = T(0) - \frac{K_2'}{\alpha C}$, and $T(0)$ is the initial temperature at $t = 0$. Further manipulate Eqn. (6) using integration by part,

$$\begin{aligned} T(t) =& \frac{K_2'}{\alpha C} + \frac{1}{\alpha C}\sum_x (G_x \cdot g_x(t)) \\ &+ C_0' e^{-\alpha t} - \frac{1}{\alpha C} e^{-\alpha t}\sum_x (G_x \int e^{\alpha t} g_x'(t)dt). \end{aligned} \quad (7)$$

The steady state temperature of m, T_{ss}^m, can then be represented as, assuming temperature change rate $g_x'(t) \to 0$ and $e^{-\alpha t} \to 0$ (as $t \to \infty$),

$$T_{ss} = \frac{K_2'}{\alpha C} + \frac{1}{\alpha C}\sum_x (G_x \cdot g_{x,ss}), \quad (8)$$

where $g_{x,ss}$ is the neighbor x's steady state temperature. To calculate the steady state temperature for processors and heat spreader, the system of equations (8) can be conveniently solved in the matrix form.

The thermal timing constraint, d_i^{thr}, is rigorously defined if $T_{ss}^m > \mathcal{T}$, where

$$\begin{aligned} \mathcal{T} =& \frac{K_2'}{\alpha C} + \frac{1}{\alpha C}\sum_x (G_x \cdot g_x(d_i^{thr})) + C_0' e^{-\alpha d_i^{thr}} \\ &- \frac{1}{\alpha C} e^{-\alpha d_i^{thr}} \sum_x (G_x \int_0^{d_i^{thr}} e^{\alpha t} g_x'(t)dt). \end{aligned} \quad (9)$$

Solving (9) at runtime has prohibitive complexity because $g_x(t)$ is difficult to be profiled/projected accurately in multiprocessors with quick thermal conduction. We begin from a simple scenario and make safe assumptions for more complicated cases. First of all, we study the scenario that, at the time of deciding d_i^{thr} on m, no task activation on other processors exists, namely $g_x(t)$ rise of all processors are solely due to task t_i. We assume that $g_x(t)$ rises as a step function from $g_x(0)$ to steady state $g_{x,ss}$. This assumption is conservative by making temperature increase faster. Plugging in the step function of $g_x(t)$, Eqn. (9) becomes

$$\frac{\mathcal{T} - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum_x (G_x \cdot g_{x,ss})}{T(0) - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum_x (G_x \cdot g_{x,ss})} = e^{-\alpha d_i^{thr}}. \quad (10)$$

$$d_i^{thr} = -\frac{1}{\alpha}\ln \frac{\mathcal{T} - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum_x (G_x \cdot g_{x,ss})}{T(0) - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum_x (G_x \cdot g_{x,ss})}. \quad (11)$$

If for some processor x, $g_{x,ss} > \mathcal{T}$, we replace $g_{x,ss}$ with $\mathcal{T}$.

As the more generic case, assume some tasks (t_x) start earlier and run concurrently with t_i. We define t_x as a "hot" ("cool") task if $g_{x,ss} > g_x(0)$ ($g_{x,ss} \le g_x(0)$), where $g_x(0)$ is instantaneous temperature at $t = 0$. The cool task can be caused by decreasing the voltage/frequency (v_x, f_{v_x}) level, or ceased execution. For the hot task, the assumption of a step-up $g_x(t)$ is still conservative as explained above. However, for the cool task, assuming the step-down $g_x(t)$ leads to relaxation of d_i^{thr} approximation, because faster dropping $g_x(t)$ could lead to slower rising temperature of hot tasks in the system, and effectively results in longer d_i^{thr} using (11), jeopardizing the conservativity of d_i^{thr}. On the other extreme, assuming the non-step-down, namely, a constant temperature $g_x(t) = g_x(0)$, is over-conservative. We can then apply a discount, namely $w_x g_{x,ss}$, as the compromise. The discount factor w_x should be empirically derived, e.g. based on the past observation of neighbor impact [8]. Then, for $g_x(t)$ in (9), we choose step functions for hot neighbors $\tilde{h}$, and discounted step functions for cool neighbors $\tilde{c}$. The thermal timing constraint is then,

$$d_i^{thr} = -\frac{1}{\alpha}\ln \frac{\mathcal{T} - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum (G_{\tilde{h}} g_{\tilde{h},ss} + w_{\tilde{c}} G_{\tilde{c}} g_{\tilde{c},ss})}{T(0) - \frac{K_2'}{\alpha C} - \frac{1}{\alpha C}\sum (G_{\tilde{h}} g_{\tilde{h},ss} + w_{\tilde{c}} G_{\tilde{c}} g_{\tilde{c},ss})}. \quad (12)$$

Last but not least, we do not consider any task activation/deactivation happening later. Because at the time of their activation/deactivation, it falls into the scenario considered above.

V. Cycle Maximization Using Guided-Search

We present the heuristic for total cycle maximization, $\sum_{t_i\in\mathcal{N}} c_i$, given constraints under the real-time embedded scenario, including deadline constraint d_i, system energy constraint $E_{\mathcal{N}}$, and the thermal timing constraint d_i^{thr} derived above. The task set $\mathcal{N}$ contains parallel tasks/workloads, t_i, which are to be dispatched and run on heterogeneous parallel processors. The optimization is formulated as,

Maximize

$$\sum_{t_i\in\mathcal{N}} c_i \quad (13)$$

Subject to

$$\frac{c_i}{f_i} \le min\{d_i, d_i^{thr}\}, \forall t_i \in \mathcal{N} \quad (14)$$

$$\sum_{t_i\in\mathcal{N}} (c_i E_{f_i}^{cyc}) \le E_{\mathcal{N}} \quad (15)$$

where (14) indicates the timing constraint, namely the deadline due to performance or thermal requirement, whichever is shorter. The energy constraint is indicated in (15), where $E_{f_i}^{cyc}$ is the per-cycle energy, ref. (2). We assume the above constraints are values after substraction from worst case timing and energy overheads due to frequency transition, namely transiting from highest to lowest voltage/frequency levels or vice versa. The goal is to find appropriate f_i for each $t_i \in \mathcal{N}$, such that (13) can be maximized.

Applying existing solutions for the problem has been proved to have a complexity of NP-completeness [18]. To efficiently solve the optimization problem, we propose a heuristic to determine t_i's frequency scaling direction, namely, increasing or decreasing f_i. The frequency scaling decisions are guided by ensured total cycle increment under each successful search for frequency scaling directions, so we name the heuristic as guided-search, which is based on the following observations,

- #1 Scaling up f_i requires higher power consumption, leading to higher T_{ss}, and decreases d_i^{thr} if available. Similarly, scaling down f_i increases d_i^{thr}.
- #2 The per-cycle energy, E_i^{cyc}, as defined in (2), is a function of both T and f_i under v_i. Note from above that T changes following f_i; however, empirical data shows that T changes in a smaller extent compared to f_i. Hence, according to (2), we assume that E_i^{cyc} is monotonically decreasing with f_i.
- #3 Non-ideal selection of f_is for $\mathcal{N}$ fails to fully utilize timing and energy resources simultaneously, and can have either (14) or (15) equalized but not both.
- #4 Assuming (15) is equalized but some t_i in (14) is not, we can select a t_i and scale *up* f_i (hence decrease $E_{f_i}^{cyc}$). To keep (15) equal, c_i can be increased. According to (14), c_i increase can be realized under increased f_i. At the mean time, f_i increase could reduce d_i^{thr} due to increased temperature, preventing c_i increase. Thus, deciding whether f_i can be increased is subject to whether c_i is actually increased after the scaling.
- #5 If all (14) are equalized but (15) is not, we can select a t_i and scale *down* f_i to increase $E_{f_i}^{cyc}$. By scaling down f_i, the d_i^{thr} can be extended, producing space for c_i increase. At the mean time, the c_i and $E_{f_i}^{cyc}$ increase should not violate (15).

Observations #4 and #5 reveal the frequency scaling directions to increase c_i in both situations. Thus, $\sum c_i$ can be steadily increased in a guided search process. As the first step, we have to initially reach one of the two situations (#4 and #5). We assume that each t_i has been assigned c_i^{min} cycles before runtime. The initial f_i is set to the highest value. By maximizing f_i, for each t_i, l.h.s of (14) is expected to be less than the r.h.s.. According to #1, it might happen that d_i^{thr} is shortened such that $\frac{c_i}{f_i} > d_i^{thr}$ for some t_i. In this case, f_i is reduced until $\frac{c_i}{f_i} \leq d_i^{thr}$. If the violation persists at the lowest f_i, we continue to scale down other task's frequencies starting from the one with the highest steady state temperature, in the hope that d_i^{thr} becomes longer due to system-wide cooling down. Otherwise an anomaly should be reported and handled by the scheduler.

TABLE I: HotSpot setup and core parameters

	100MHz	300MHz
Dyn.	821.2mW	2532mW
Lkg.	$P = 4T + 695$mW	
amb. temp		45°C
T. size (mm^2)		8.5×8.5

By scaling up the f_i, energy constraint (15) is also expected to be satisfied given the monotonically decreasing relationship between $E_{f_i}^{cyc}$ and f_i. After all the constraints of (14) and (15) are met, we choose to increase the c_i of one or more t_i until the equality of either (14) or (15) is reached, whereby situation #4 or #5 happens.

Under situation #4, f_i in (14) is scaled up by one level to f_{i+}, while c_i should be increased to maintain equality of (15). The choice of t_i depends on how much thermal violation can be avoided during f_i scaling up. We select the t_i with the lowest steady state temperature in the system as calculated by (8). In the process of frequency reduction, it may happen that all branches in (14) are equalized due to increased c_i, but there are still unused energy in (15). In this case, situation #5 occurs and we scale down f_i by one level to f_{i-}, while c_i might still be increased, due to extended d_i^{thr}, to maintain the equality of (14). The process continues until #4 occurs again, or termination conditions happen.

The frequency scaling process terminates under any of the three conditions:

- $c_i = c_i^{max}, \forall t_i \in \mathcal{N}$;
- when scaling up f_i is required, all f_is are in the highest allowable frequency;
- when scaling down f_i is required, all f_is are in the lowest allowable frequency.

VI. Experimental Results

We use the HotSpot [15] thermal simulation tool as the evaluation platform on the performance of our algorithm. We assume the platform contains 9 tiled FFT cores synthesized on the Xilinx Virtex-6 XC6VLX240T board. Table I shows the tile data necessary to derive the thermal profile. The power values of the FFT cores are derived using ModelSim based on the Virtex implementation. Under nano-scale device sizes, the instantaneous leakage power is dependent on the device temperature. According to the XPower Analyzer, the leakage power is linear to the temperature as shown in Table I.

To evaluate the performance of our algorithm, we synthesized a set of pseudo adaptive workloads on the platform. The number of workloads varies between [8, 40] in the step of 4. The workloads are assumed to be inter-dependent in execution sequence, which is randomly generated using TGFF [19]. The execution is based on a slack claim and redistribution fashion [18], where the preceding task, t_p, finishes 30% before

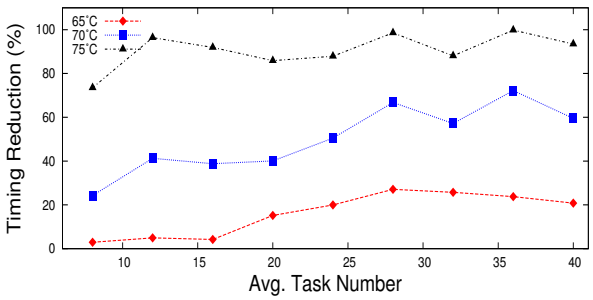

Fig. 4: Percentage of thermal timing constraints to normal deadlines.

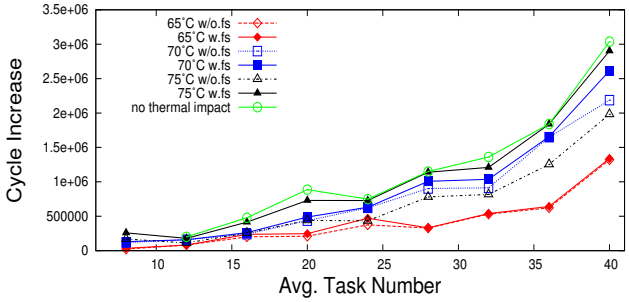

Fig. 5: Cycle increase of the proposed algorithm compared to no frequency scaling.

its statically determined time. The task group, denoted as $\mathcal{N}$ in Section III, then adopts the slack timing and associated energy. In constraint (14), the d_i is the sum of t_i's and 30% of t_p's execution time, and in (15) $E_{\mathcal{N}}$ is the saved energy associated with 30% of t_p.

To reflect the effectiveness of adaptive workload on the temperature constraint, we carefully adjust the workload deadlines to be compatible with the thermal timing. Fig. 4 shows how temperature constraint shortens the deadline for the set of workloads under test, ranged between 8 and 40. By reducing the temperature threshold from 75°C to 70°C and 65°C, the average deadline shortening rate (the ratio between d_i^{thr} and d_i, ref. Eqn. (14)) reduces from 90.6% to 50.0% and 16.1% respectively. The average cycle gain, as shown in the solid lines in Fig. 5, reduces accordingly by 14.9% and 58.2%. However, compared to not applying the frequency scaling approach, which is plotted in the dotted lines in Fig. 5 and derived by setting all frequencies to the middle of the available frequency levels, our approach gains on average 31.5%, 9.5%, and 3%, under temperature constraints 75°C, 70°C, and 65°C, respectively. The difference in cycle gain under different temperature constraints, can be explained such that more timing resources at higher temperature threshold are provided for frequency scaling. Note that large reduction in timing constraints, as shown in Fig. 4, does not necessarily leads to large reduction in cycle gain in Fig. 5. This is because the experiment is based on the runtime slack claiming and distribution, and this phenomenon coincides with the finding in [18], where parallel energy constraint is more of the scarce resource rather than timing.

VII. Conclusions

In this paper, we present a frequency scaling algorithm that targets adaptive workloads, which is employed as an application-level thermal management scheme. The performance of the algorithm, provided with accuracy of thermal sensing data, is as large as 31.5% compared to not applying frequency scaling. As an immediate future work, we would like to synthesize real-life systems, instead of pseudo workloads, for more concrete evaluation of the algorithm. Meanwhile, a runtime FPGA-based core relocation scheme, as a joint and complementary design perspective to adaptive workloads, is being studied for more efficient thermal control.

References

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Trans. on Circuits Syst. Video Techn.*, vol. 17, no. 9, pp. 1103–1120, 2007.

[2] T. Acharya and P. S. Tsai, "Jpeg2000 standard for image compression: Concepts, algorithms and vlsi architectures," Wiley 2004.

[3] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," 2000.

[4] T. Sato, J. Ichimiya, N. Ono, K. Hachiya, and M. Hashimoto, "On-chip thermal gradient analysis and temperature flattening for soc design," in *ASP-DAC*, 2005, pp. 1074–1077.

[5] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Trans. CAD*, vol. 24, no. 7, pp. 1042–1053, 2006.

[6] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *DAC*, 2008, pp. 734–739.

[7] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Proactive temperature balancing for low cost thermal management in mpsocs," in *ICCAD*, 2008, pp. 250–257.

[8] G. Liu, M. Fan, and G. Quan, "Neighbor-aware dynamic thermal management for multi-core platform," in *DATE*, 2012, pp. 187–192.

[9] M. Bao, A. Andrei, P. Eles, and Z. Peng, "Online thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration," in *DAC*, 2009, pp. 490–495.

[10] Y. Ge and Q. Qiu, "Dynamic thermal management for multimedia applications using machine learning," in *DAC*, 2011, pp. 95–100.

[11] S. Zhang and K. S. Chatha, "Thermal aware task sequencing on embedded processors," in *DAC*, 2010, pp. 585–590.

[12] H. Huang, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," in *DAC*, 2011, pp. 363–368.

[13] V. Hanumaiah and S. B. K. Vrudhula, "Reliability-aware thermal management for hard real-time applications on multi-core processors," in *DATE*, 2011, pp. 137–142.

[14] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *IEEE Trans. VLSI Syst.*, vol. 19, no. 10, pp. 1884–1897, 2011.

[15] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: A compact thermal modeling methodology for early-stage vlsi design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 5, 2006.

[16] V. K. V. *et al.*, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC*, 2013, pp. 1–9.

[17] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *DATE*, 2007, pp. 1526–1531.

[18] H. Yu, Y. Ha, and B. Veeravall, "Quality-driven dynamic scheduling for real-time adaptive applications on multiprocessor systems," *IEEE Trans. on Computers*, vol. 62, no. 10, pp. 2026–2040, 2013.

[19] "Tgff: task graphs for free," 2008, http://ziyang.eecs.umich.edu/~dickrp/tgff/.